第 39 卷第 8 期
录用定稿

计算机应用研究
Application Research of Computers

Vol. 39 No. 8
Accepted Paper

折扣{0-1}背包问题粒子群算法的贪婪修复策略探究 *

代祖华，周　斌，龙玉晶，王宗泉

(西北师范大学 计算机科学与工程学院，兰州 730070)

摘　要：群智能启发式算法求解折扣{0-1}背包问题(D{0-1}KP)时，为提升求解效率和求解质量，需采用某种修复与优化策略将非正常编码个体转换为符合解约束条件的编码个体。在引入项集价值密度概念基础上，以粒子群算法(PSO)为例，提出一组基于项集的贪婪修复与优化方法(Group Greedy Repair and Optimization Algorithm，GGROA)，并进一步构造 PSO-GGRDKP 算法(PSO based GGROA for solving D{0-1}KP)以探究 GGROA 方法的可行性和性能。PSO-NGROADKP(PSO based NGROA for solving D{0-1}KP)和 PSO-GRDKP(PSO based GROA for solving D{0-1}KP)是基于项贪心修复与优化方法的粒子群算法。在 D{0-1}KP 标准数据集的实验结果表明：与 PSO-NGROADKP 和 PSO-GRDKP 相比，PSO-GGRDKP 算法的解误差率略高，但算法时间性能分别提升 13.8%、12.9%。

关键词：折扣{0-1}背包问题；启发式算法；粒子群算法；非正常编码个体；贪心修复与优化；D{0-1}KP 数据集

中图分类号：TP391　　doi: 10.19734/j.issn.1001-3695.2021.12.0701

Greedy repair strategy of particle swarm optimization for discounted {0-1} knapsack problem

Dai Zuhua, Zhou Bin, Long Yujing, Wang Zongquan

(*College of Computer Science&Engineering*, *Northwest Normal University Gansu* 730070, *China*)

Abstract: Swarm intelligence heuristic algorithm is used to solve discounted {0-1} knapsack problem (D{0-1} KP) . In order to improve the solution efficiency and quality, a repair and optimization strategy is needed to convert abnormal coding individuals into coding individuals that meet the solution constraints. On the basis of introducing the concept of group value density, taking particle swarm optimization algorithm (PSO) as an example, a set of greedy repair and optimization methods based on group (GGROA) is proposed, and the PSO based GGROA for solving D{0-1}KP algorithm (PSO-GGRDKP) is further constructed to explore the feasibility and performance of GGROA. PSO-NGROADKP and PSO-GRDKP are PSO algorithms based on item greedy repair and optimization method. The experimental results on D {0-1} KP standard data set show that compared with PSO-NGROADKP and PSO-GRDKP, PSO-GGRDKP has slightly higher error rate, but the time performance of the algorithm is improved by 13.8% and 12.9% respectively.

Key words: discount {0-1} knapsack problem; heuristic algorithm; particle swam optimization algorithm; non-normal coding individual; greedy repair and optimization; D{0-1}KP dataset

0　引言

{0-1}背包问题({0-1}Knapsack Problem, {0-1}KP)是计算机科学一个重要的 NP Complete 问题，也是一类经典组合优化问题，在商业、经济、管理、安全等领域有着广泛应用背景。{0-1}KP 自被提出后的几十年里被反复地研究，衍生出精确算法、非精确算法两大类算法。精确算法有动态规划、回溯法和分支限界法等；非精确算法主要有随机算法、近似算法、生物算法等。2005 年，由 GUDER 首次提出的折扣{0-1}背包问题(Discount {0-1} Knap-sack Problem, D{0-1}KP)[1,2]是{0-1}背包问题的拓展形式，作为刻画折扣销售、捆绑销售等商业活动现象的经典数学模型，是商家设计商业营销方案、消费者购买商品决策的科学计算依据。

D{0-1}KP 实例由一组项集(item_group)组成，每个项集有 3 项物品(项，item)可供背包装入选择，定义 1 给出了 D{0-1}KP 第一数学模型。

定义 1　D{0-1}KP 第一数学模型[3]：给定 n 个项集和载重为 C 的背包，每个项集 $i(i=0,1,\cdots,n-1)$ 由 3 个项(item)组成，编号记作 $3i,3i+1,3i+2$，项对应重量系数记作 w_{3i},w_{3i+1},w_{3i+2}，各项对应价值系数记作 p_{3i},p_{3i+1},p_{3i+2}，其中 $p_{3i+2}=p_{3i}+p_{3i+1}, w_{3i+2}<w_{3i}+w_{3i+1}$，$w_{3i}<w_{3i+1}<w_{3i+2}, w_{3i+2}\le C$，$\sum_{i=0}^{n-1}w_{3i+2}>C, p_j, w_j(0\le j\le 3n-1), C$ 都是正整数，项集 i 的各项价值密度记作 $e_{3i},e_{3i+1},e_{3i+2}=p_{3i}/w_{3i}, p_{3i+1}/w_{3i+1}, p_{3i+2}/w_{3i+2}$，每个项集至多有一项被选择装入背包。在不超过背包载重量 C 的条件下，从给定项集选择满足装入背包要求的项，使得装入背包所有项的价值系数之和达到最大。

D{0-1}KP 是一个特殊整数规划问题，定义决策变量 $x_j=1(0\le j\le 3n-1)$ 表示项 j 装入背包，$x_j=0$ 表示项 j 未装入背包，对于决策向量 $(x_0,x_1,\cdots,x_{3n-1})$，D{0-1}KP 的整数规划模型为

$$\max f(x)=\max\sum_{i=0}^{n-1}(x_{3i}p_{3i}+x_{3i+1}p_{3i+1}+x_{3i+2}p_{3i+2}) \quad (1)$$

其中：

$$x_{3i}+x_{3i+1}+x_{3i+2}\le 1, x_{3i},x_{3i+1},x_{3i+2}\in\{0,1\} \quad (2)$$

$$\sum_{i=0}^{n-1}(x_{3i}w_{3i}+x_{3i+1}w_{3i+1}+x_{3i+2}w_{3i+2})\le C \quad (3)$$

D{0-1}KP 的项集有 4 种选择状态，如果背包容量和项的价值系数取值范围很大，则不选择某个项集的约束条件较难确定，这意味着 D{0-1}KP 的算法难度要大于{0-1}KP[4]。以{0-1}KP 研究成果为基础，学者们进一步研究了 D{0-1}KP 的各

收稿日期：2021-12-18；**修回日期：**2022-03-11　**基金项目：**兰州市科技发展指导性计划项目(2020-ZD-136)；西北师范大学研究生培养与课程改革项目(2020KGLX01009)；国家自然科学基金资助项目(61762080)

作者简介：代祖华(1971-)，女，甘肃榆中人，副教授，硕导，博士，主要研究方向为组合优化理论与算法、深度强化学习、自然语言处理(2812704000@qq.com)；周斌(1996-)，男，湖南澧县人，硕士研究生，主要研究方向为组合优化理论与算法、深度强化学习；龙玉晶(1997-)，女，湖南邵东县人，硕士研究生，主要研究方向为组合优化理论与算法、深度强化学习；王宗泉(1998-)，男，河南范县人，主要研究方向为组合优化理论与算法、深度强化学习.

类算法。其中基础动态规划算法[5](Basic Dynamic Programming, BDP)是处理小规模数据的精确解算法。2016 年贺毅朝等人[4]以“所选择项的价值系数之和使总重量最小”原则构造动态规划目标函数，提出一种新动态规划算法(New Exact algorithm for D{0-1}KP, NE-DKP)，当背包容量大于所有的项集第三项价值累加和时，NE-DKP 算法性能好于 BDP 算法。动态规划算法是伪多项式时间复杂度，一般不适用于大规模 D{0-1}KP 实例的求解。

群智能启发式算法是近年来求解 D{0-1}KP 的主流算法，对于解决大规模实例有突出性能优势。应用 D{0-1}KP 第一数学模型设计启发式算法，决策向量 $(x_0,x_1,\cdots,x_{3n-1})$ 的不同取值组合对应不同个体，这种二元编码法虽然便于个体演化算子的实现，但由于可行解约束条件(2)(3)的限制，编码空间中非正常编码个体(即个体编码不对应问题可行解)的概率至少为 $1-(1/2)^n$ [3]。为此，需要采用某种策略将非正常编码个体转换为符合解约束条件的编码个体，以提升求解效率和求解质量。Michalewicz[6]、贺毅朝[3,4]等学者先后提出的贪心修复与优化策略是消除非正常编码个体效果较好的方法。文献[4]在研究 D{0-1}KP 粒子群求解算法时，证明了给定项集中三个项目的价值重量比率关系只有四种情况，利用这一特性提出了 GR-DKP(Greedy Repair Algorithm for D{0-1}KP)算法，设计了基于 GR-DKP 的粒子群求解算法(the PSO based Greedy Repair Algorithm for solving D{0-1}KP，PSO-GRDKP)[4]。文献[3]在研究 D{0-1}KP 第一遗传算法(First Genetic Algorithm，FirEGA)中再次提出贪心修复与优化算法(Greedy Repair and Optimization Algorithm, GROA)[3]，GROA 算法与 GR-DKP 算法均按照非递增项价值密度对项进行贪心选取，若项集中有多项是选取状态时，选择价值密度最大项。杨洋等人进一步优化了 GROA 算法，构造了新贪心修复优化算法(New Greedy Repair and Optimization Algorithm，NGROA)[7]，该算法也按照非递增项价值密度对项进行贪心选取，但当项集中有多项是选取状态时，选择项集的价值最大项，应用 NGROA 算法提出 NFirEGA(New First Genetic Algorithm)的研究结果表明较之 FirEGA 提升了 D{0-1}KP 求解质量。最近，文献[8]定义了项集价值密度概念，提出按非递增项集价值密度对项进行贪心选取的策略，当项集中多项是选取状态时，选择满足解约束条件的价值密度最大项。上述贪婪修复与优化算法[3~8]的时间复杂度为 $O(n)$ 。多数研究者采用文献[3]提出的 GROA 算法设计 D{0-1}KP 的各类启发式进化算法，如差分进化算法(differential evolution algorithm，DE)[9]、变异蝙蝠算法(Mutated Double Codes Binary Bat Algorithm，MDBBA)[10]、差分进化帝王蝶优化启发式算法(Monarch Butterfly Optimization with Differential Evolution，DEMBO)[11]、飞蛾搜索算法(Moth Search Algorithm，MS)[12] 、基于 Lagrange 插值的学习猴群算法(Lagrange Interpolation based Learning Monkey Algorithm，LSTMA)[13]、基于环论的演化算法 (Ring Theory Based Evolutionary Algorithm，RTEA)[14]、基于离散混合教学的优化算法(Discrete Hybrid Teaching Learning based Optimization Algorithm，HTLBO)[15]等，以上研究常选用 FirEGA 算法[3]和基本启发式算法作为基线，通过对比求解质量和收敛速度以论证所研究算法的优化性能。

在采用第一数学模型和 GROA 算法构造 D{0-1}KP 启发式算法的研究中[10,11,14]，文献[10]的仿真实验结果表明，在 UDKP 实例和 SDKP 实例上，双重编码二进制蝙蝠算法(Double codes Binary Bat Algorithm，DBBA)的求解质量比 FirEGA 差，在 IDKP 实例和 WDKP 实例上，DBBA 的求解质量好于 FirEGA；帝王蝶优化算法(Monarch Butter fly Optimization，MBO)在 best、mean 及 worst 三个评价指标上的求解结果要明显差于 FirEGA 算法[11]；PSO-GRDKP 算法求解精度和稳定性优于 FirEGA[14]。这些研究结果表明，采用同一类贪婪修复与优化方法的不同启发式算法求解性能会有差异。本文基于项集价值密度[8]概念，构造一组基于项集的贪婪修复和优化方法，并探究其可行性和性能，进一步讨论贪婪修复方法与数据实例类型之间的适应性关系。为避免启发式算法差异对研究内容的影响，以粒子群算法为例来构造 D{0-1}KP 求解算法。

1 相关研究工作

1.1 二元粒子群优化算法

Kennedy 和 Eberhart 通过对鸟群捕食行为的研究，在 1995 年提出标准粒子群算法(Particle Swarm Optimization，PSO)[16]，BPSO(Binary Particle Swarm Optimization)是一种应用于离散空间搜索的二元粒子群优化算法[17]，Bansal 和 Deep 用粒子速度作为{0-1}KP 中物品选择为 1 或 0 的概率，提出了一种修正二元粒子群优化(Modified Binary Particle Swarm Optimization，MBPSO) 算法来解决背包问题[18]。MBPSO 算法原理描述为：假设搜索空间为D 维，定义D 维向量 $x_i=(x_{i1},x_{i2},\cdots,x_{ij},\cdots,x_{iD})$ 表示粒子群中第 i 个粒子的位置，对应粒子速度为 $v_i=(v_{i1},v_{i2},\cdots,v_{ij},\cdots,v_{iD})$ ，个体极值 $p_{best}=(p_{best1},p_{best2},\cdots,p_{bestj},\cdots,p_{bestD})$ 表示粒子群某次迭代的最优解，全局极值 $g_{best}=(g_{best1},g_{best2},\cdots,g_{bestj},\cdots,g_{bestD})$ 表示种群在进化过程中的全局最优解，个体极值和全局极值由适应度函数确定，粒子进化公式为

$$v_{ij}^{t+1}=v_{ij}^{t}+c_1\cdot r_1\cdot(p_{bestj}^{t}-x_{ij}^{t})+c_2\cdot r_2\cdot(g_{best}-x_{ij}^{t}) \tag{4}$$

$$sig(v_{ij}^{t+1})=\frac{1}{1+e^{-v_{ij}^{t+1}}} \tag{5}$$

$$x_{ij}^{t+1}=\begin{cases}0, & if\ r_3\ge sig(v_{ij}^{t+1})\\ 1, & otherwise\end{cases} \tag{6}$$

其中： t 示进化迭代次数， v_{ij}^{t} 表示在 t 次进化迭代中第 i 个粒子的第 j 维速度， x_{ij}^{t} 表示在 t 次进化迭代中第 i 个粒子的第 j 维位置。(4)式为粒子群的进化动力方程，主要由三方面构成： v_{ij}^{t} 属于粒子个体的惯性势； $c_1\cdot r_1\cdot(p_{bestj}^{t}-x_{ij}^{t})$ 来自粒子个体当前历史最好位置 p_{best} 的引力势，代表粒子“个体认知”； $c_2\cdot r_2\cdot(g_{best}-x_{ij}^{t})$ 是来自群体当前历史最好位置 g_{best} 的引力势，代表粒子“社会认知”[19]。学习因子 c_1,c_2 作为粒子“个体认知”和“社会认知”的主要加权系数，表示进化过程对粒子个体信息和社会信息的学习继承程度，主要影响着粒子的优化目标识别能力。 r_1,r_2,r_3 表示(0, 1)之间的随机数。

PSO-GRDKP 算法[4]是 D{0-1}KP 的一种粒子群求解算法，算法搜索空间维度 D=3n−1，粒子适应度函数对应个体解的背包装入价值，学习因子 c_1,c_2 的取值是 2，算法时间复杂度是 $O(n^3)$ 。该算法利用贪婪修复与优化算子对种群中不可行粒子进行校正和优化，有效提升种群中优质个体比率，增强算法寻优能力。

1.2 不可行个体贪婪修复与优化算法

何毅朝等人在研究 D{0-1}KP 问题时先后提出 GROA[3] 、GR-DKP[4] 两种贪婪修复算法，其中 GR-DKP 在研究粒子群算法(PSO-GRDKP)时提出，以下给出 GR-DKP 算法伪代码。

算法 1 GR-DKP

输入： $3n$ 个项的价值向量 P 、重量向量 W ；背包容量 C ； $H_{[0,\cdots,3n-1]}$ ：按非递增项价值密度次序保存各项下标； $x_{[0,\cdots,3n-1]}$ ：待修复粒子。

输出：修复优化后的可行粒子 $x_{[0,\cdots,3n-1]}$ 和适应度值 $f(x)$ 。

1. $fweight=0, fvalue=0$
2. FOR i IN $range(0,n-1)$ ：
    IF $(x_{3i}+x_{3i+1}+x_{3i+2}\ge 1)$ ：
      $x_{3i}=0, x_{3i+1}=0, x_{3i+2}=0$
      $j=\arg\max_{j\in\{3i,3i+1,3i+2\}}\left(p_j/w_j\right)$
      $x_j=1, fweight+=w_j$
3. $k=3n-1$
4. WHILE $weight>C$ AND $k\ge 0$:
    IF($x_{H[k]}=1$):
      $fweight-=w_{H[k]}, x_{H[k]}=0$

$k = k - 1$

5.　FOR j IN $range(0, 3n-1)$:

$i = \text{int}\left(H[i]/3\right)$

IF($x_{3i} + x_{3i+1} + x_{3i+2} == 0$ AND $fweight + w_{H[j]} \le C$):

$x_{H[j]} == 1,\ fweight += w_{H[j]}$

6.　FOR i IN $range(0, 3n-1)$: $fvalue += x_i \times p_i$

7. RETURN $x, fvalue$

算法 1 第 2~4 步是对粒子 x 的修复操作、第 5 步是对粒子 x 的优化操作，算法时间复杂度是 $O(n)$ 。将算法 1 第 2 步的 $j = \arg\max_{j\in\{3i,3i+1,3i+2\}}\left(p_j/w_j\right)$ 置换为 $j = \arg\max_{j\in\{3i,3i+1,3i+2\}}(p_j)$ ，则算法演化为NGROA[7]。

2　基于项集的 D{0-1}KP 粒子群优化算法

2.1　基于项集的粒子修复与优化算法贪婪策略

D{0-1}KP 已有研究提出的不可行粒子贪婪修复策略[3,4,7]，均按照项价值密度大小对数据排序预处理。本文引入项集价值密度 $R_i (i \in \{0,1,\cdots,n-1\})$ ，按照 R_i 大小以项集为单位对数据进行非递增排列，之后按次序选取项集的项。文献[8]给出了三种 R_i 算法如下：

$$R_i^1 = \max(e_{3i}, e_{3i+1}, e_{3i+2}) \tag{7}$$

$$R_i^2 = \sum_{k=0}^{2} e_{3i+k} \tag{8}$$

$$R_i^3 = \frac{\sum_{k=0}^{2} p_{3i+k}}{\sum_{k=0}^{2} w_{3i+k}} \tag{9}$$

易见， R_i^1 算法等效于 GR-DKP[4]策略，结合 NGROA 算法思想，定义 R_i^4 如下：

$$R_i^4 = e_{3i+2} \tag{10}$$

以下给出项集 i 两种项选择策略 $item_i$ 下：

$$item_i^1 = \max_j \{e_j \mid x_j = 1, j \in \{3i, 3i+1, 3i+2\}\} \tag{11}$$

$$item_i^2 = \begin{cases} 3i+2, & \text{如果 } x_{3i+2} = 1 \\ \max_j \{e_j \mid x_j = 1, j \in \{3i, 3i+1\}\}, & \text{其他} \end{cases} \tag{12}$$

合式(7)~(10)和式(11)(12)，得到八种 D{0-1}KP 基于项集的贪心修复优化方法(Group Greedy Repair and Optimization Algorithm, GGROA)，见表 1 所示。

表 1　D{0-1}KP 粒子群算法的 GGROA 策略

Tab. 1　GGROA of D {0-1} KP for PSO algorithm

项集价值密度 $R_i, i \in \{0,\cdots,n-1\}$	R_i^1	R_i^2	R_i^3	R_i^4
项集 i 的项选择策略	$item_i^1$		$item_i^2$	

GGROA 的伪代码描述见算法 2，参数 $m \in \{1,2,3,4\}$ 对应式(7)~(10)的四种项集价值密度，参数 $z \in \{1,2\}$ 对应式(11)(12)的两种项选择策略。

算法 2　GGROA(H^m, $item^z$)

输入： $3n$ 个项的价值向量 P 、重量向量 W ；背包容量 C ； $H^m_{[0,\cdots,n-1]}$ ：按非递增项价值密度次序保存各项下标； $x_{[0,\cdots,3n-1]}$ ：待修复粒子。

输出：修复优化后的可行粒子 $x_{[0,\cdots,3n-1]}$ 和适应度值 $f(x)$ 。

1: FOR i IN $range(0, n-1)$:

$Flag_i = 0$

2:　$fweight = 0,\ fvalue = 0$

3:　FOR i IN $range(0, n-1)$:

IF $(x_{3i} + x_{3i+1} + x_{3i+2} \ge 1)$:

$x_{3i} = 0, x_{3i+1} = 0, x_{3i+2} = 0$

$j = item_i^z$

$x_j = 1,\ fweight += w_j, Flag_i = 1$

4:　$k = n - 1$

5:　WHILE $fweight > C$ AND $k \ge 0$:

IF($Flag_{H^m[k]} == 1$):

$j = item_i^{H^m[k]}$

$fweight -= w_j, x_j = 0, Flag_{H^m[k]} = 0$

$k = k - 1$

6:　FOR j IN $range(0, n-1)$:

$j = item_i^{H^m[k]}$

IF($Flag_{H^m[k]} == 0$ AND $fweight + w_j \le C$):

$x_j == 1,\ fweight += w_j$

7:　FOR i IN $range(0, 3n-1)$: $fvalue += x_i \times p_i$

8: RETURN $x, fvalue$

算法 2 第 2 步~第 5 是对粒子 x 的修复操作、第 6 步是对粒子 x 的优化操作，算法时间复杂度是 $O(n)$ 。

2.2　基于 GGROA 的 D{0-1}KP 粒子群优化算法

文献[4]在构造 PSO-GRDKP 算法中，采用了基础离散粒子进化式(4)~(6)，学习因子的取值 c_1, c_2 是 2。为避免粒子进化公式和进化参数对研究内容的影响，方便同一基准条件下同类研究间的辨析，以下采用文献[4]的方案构造 PSO-GGRDKP(PSO based GGROA for solving D{0-1}KP)，伪代码见算法 3，其中参数 m, z 算法 2。

算法 3　PSO-GGRDKP(m, z)

输入： $3n$ 项的价值向量 P 、重量向量 W ，背包容量 C ，种群规模 N ，进化代数 T ，学习常数 $c_1 = c_2 = 2$

输出：最佳粒子 g_{best} 和装入背包总价值的近似最优解 $fitness(g_{best})$ 。

1: FOR i IN $range(0, n-1)$:

$R_i^m = function_m(P_i, W_i)$

2:　排列 $R_i^m \ge R_{i+1}^m, i \in \{0,\cdots,n-1\}$ ，按项集价值密度 R_i^m 非递增次序将项集下标存入 $H^m_{[0,\cdots,n-1]}$

3:　初始化种群： $population = \{(x_i, v_i) \mid i \in (1,\cdots.N)\}$

4:　FOR i IN $range(1, N)$:

$x_i, fitness(x_i)$=GGROA(H^m, $item^z$)

5:　$best = index_\max(\{fitness(x_i) \mid i \in \{1,\cdots,N\}\})$

6:　$g_{best} = p_{best} = x_{best}$

7:　$t = 0$

8:　WHILE $t < T$:

9:　FOR i IN $range(1, N)$:

10:　FOR j IN $range(0, 3n-1)$:

$v_{ij}^{t+1} = v_{ij}^t + c_1 \cdot r_1 \cdot (p_{bestj}^t - x_{ij}^t) + c_2 \cdot r_2 \cdot (g_{bestj}^t - x_{ij}^t)$

IF $(r_3 \ge sig(v_{ij}^{t+1}))$: $x_{ij}^{t+1} = 0$

ELSE: $x_{ij}^{t+1} = 1$

11:　$x_i^{t+1}, fitness(x_i^{t+1})$=GGROA(H^m, $item^z$)

12:　IF $(fitness(x_i^{t+1}) > fitness(p_{best}))$: $p_{best} = x_i^{t+1}$

13:　IF $(fitness(p_{best}) > fitness(g_{best}))$: $g_{best} = p_{best}$

14:　$t = t + 1$

15: RETURN $g_{best}, fitness(g_{best})$

算法 3 中 r_1, r_2, r_3 是(0, 1)区间的随机数，第 1 步中 $function_m(m \in \{1,2,3,4\})$ 对应四种项集价值密度计算函数，即式(7)~10)，第 10 步 $sig(x)$ 采用式(5)，算法时间复杂度是 $O(n\log n) + 2O(n) + 2O(nN) + T \times [O(nN) + O(n)]$ ，由于 $N < n$ 且 $T < n$ ，故 PSO-GGRDKP 算法时间复杂度是 $O(n^3)$ 。

3　仿真实验与结果分析

3.1　实验方案设计

D{0-1}KP 数据集是观察和评测算法性能的标准数据集[3]，数据集由逆强相关 D{0-1}KP 实例(IDKP)、强相关 D{0-1}KP 实例(SDKP)、弱相关 D{0-1}KP 实例(WDKP)和不相关 D{0-1}KP 实例(UDKP)四类数据组成，四类 D{0-1}KP 实例的数据规模分别为 $300 \le 3n \le 3000$ 。为验证 D{0-1}KP 的各类贪婪修复策略性能特点，进行两组实验。实验一：采用八种基于项集贪婪修复优化策略的 D{0-1}KP 粒子群算法实验，以探究八种基于项集贪婪修复优化策略的可行性、性能以及与数据

实例类型的适应性关系。实验二：评测 PSO-GRDKP[4]、PSO-NGROADKP、PSO-GGRDKP 三种典型不可行个体贪婪修复方法的 D{0-1}KP 粒子群性能特点。两组实验均以比较求解时间和解计算结果分析算法性能。所有实验均在 ThinkStation P330 计算机上进行，电脑配置 Intel® Core™ i7-8700 CPU-3.2GHZ、16 GB DDR4 内存、NVIDIA P2200 显卡，操作系统是 Microsoft Windows 10 教育版，算法均使用 Python3.6 编码。

记 D{0-1}KP 实例通过基本动态规划法计算得出的最优解为 opt 、粒子群算法的粒子规模为 200、进化代数同项集数 n ，粒子群算法独立运算 20 次的近似最优解最大值为 best、平均值为 mean、最差值为 worst、平均时长为 T 。

3.2　实验一算法数据与分析

表 2 列出八种项集贪心修复策略的对应编号(id)。

表 2　D {0-1}KP 的 GGROA 策略及序号

Tab. 2　Ggroas and their serial numbers of D{0-1} KP

id	$(R_i, item_j)$	id	$(R_i, item_j)$
1	(1,1)	5	(3,1)
2	(1,2)	6	(3,2)
3	(2,1)	7	(4,1)
4	(2,2)	8	(4,2)

表 3 所示为数据规模 $3n=900$ 和 $3n=1800$ 的四类实例上运行八种贪婪修复优化方法的粒子群算法实验数据，PSO-GGRDKP 算法名称下标编号对应表 2。

应用八种贪婪修复策略对八个数据实例 IDKP3、IDKP6、SDKP3、SDKP6、UDKP3、UDKP6、WDKP3、WDKP6 分别独立求解 20 次，20 个求解结果分布情况箱线图如图 1 所示。

表 3　D{0-1}KP 标准数据实例 PSO-GGRDKP 算法实验结果

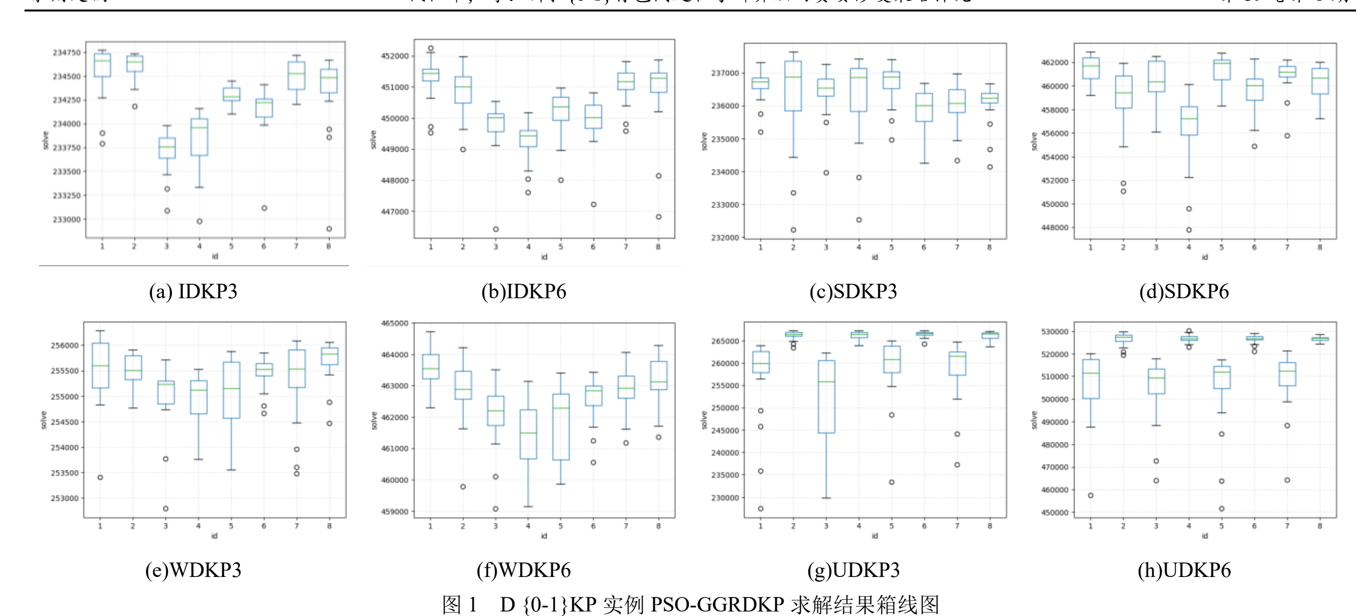
Tab. 3　PSO-GGRDKP algorithm experimental results of D {0-1} KP standard dataset

实例名(最优值)		PSO-GGRDKP$_1$	PSO-GGRDKP$_2$	PSO-GGRDKP$_3$	PSO-GGRDKP$_4$	PSO-GGRDKP$_5$	PSO-GGRDKP$_6$	PSO-GGRDKP$_7$	PSO-GGRDKP$_8$
IDKP3 (234804)	best	234772	**234737**	233983	234160	234450	234413	234720	234668
	mean	234565.65	**234592.45**	233706	233832.9	234296.05	234139.9	234490.8	234354.6
	worst	233790	**234183**	233092	232980	234103	233119	234204	232898
	T(s)	12.18	**9.07**	11.03	11.72	10.88	10.06	11.22	9.40
	ERR	0.001	**0.001**	0.005	0.004	0.002	0.003	0.001	0.002
IDKP6 (452463)	best	452250	**451982**	450537	450171	450971	450816	451829	451876
	mean	451303.05	**450804**	449725.95	449256.7	450148.5	449898.55	451071.3	450849.7
	worst	449531	**448999**	446422	447609	448009	447226	449593	446822
	T(s)	46.70	**36.34**	48.26	41.80	46.64	42.04	44.23	37.17
	ERR	0.006	**0.008**	0.013	0.011	0.010	0.012	0.006	0.012
SDKP3 (238248)	best	237313	237638	237273	237428	**237414**	236682	236973	236671
	mean	236644.85	236362.7	236428.45	236276.9	**236661.5**	235866.1	236064.6	236051.5
	worst	235198	232226	233959	232525	**234957**	234252	234342	234145
	T(s)	11.48	9.56	10.46	9.10	**10.66**	9.01	10.37	9.32
	ERR	0.01	0.01	0.01	0.01	**0.01**	0.01	0.01	0.01
SDKP6 (466097)	best	462883	461926	462498	460133	**462797**	462300	462197	462010
	mean	461454.95	458730.3	460320.15	456405.1	**461389.9**	459623.25	460858.45	460292.2
	worst	459209	451076	456090	447800	**458299**	454888	455773	457203
	T(s)	45.41	37.77	41.34	36.29	**41.70**	35.94	41.77	37.06
	ERR	0.01	0.02	0.01	0.02	**0.01**	0.01	0.01	0.01
WDKP3 (256616)	best	256286	255916	255712	255529	255879	255844	256083	**256061**
	mean	255516.15	255491.25	255003.15	254928.75	255034	255448.45	255300.9	**255713.75**
	worst	253410	254768	252796	253758	253558	254668	253481	**254466**
	T(s)	10.77	8.98	12.47	9.00	10.50	10.32	11.77	**9.85**
	ERR	0.004	0.004	0.006	0.007	0.006	0.005	0.005	**0.004**
WDKP6 (466050)	best	464720	464210	463503	463134	463414	463434	464072	**464274**
	mean	463532.4	462831.4	462020.7	461409.2	461794.25	462586.1	462883.55	**463214.55**
	worst	462308	459786	459081	459144	459866	460557	461176	**461365**
	T(s)	44.32	38.93	44.48	36.44	45.07	40.76	43.48	**37.56**
	ERR	0.005	0.007	0.009	0.010	0.009	0.007	0.007	**0.006**
UDKP3 (184006)	best	263944	**267162**	262269	267172	265006	267221	264710	267102
	mean	256631.2	**266185**	252478.55	266165.65	259174	266483.15	258556	266184.5
	worst	227502	**263431**	229837	263871	233367	264297	237289	263701
	T(s)	11.83	**10.50**	12.01	10.57	12.11	10.59	11.77	10.66
	ERR	0.05	**0.01**	0.06	0.01	0.04	0.01	0.04	0.01
UDKP6 (536578)	best	520183	**529834**	517776	530221	517264	529030	521267	528480
	mean	506575.15	**526417.6**	504295.95	526795.8	504110.5	526440.7	508569.7	526602.7
	worst	457476	**519485**	464104	522972	451696	521096	464333	524425
	T(s)	47.47	**41.07**	47.79	41.75	47.41	41.24	47.59	41.31
	ERR	0.06	**0.02**	0.06	0.02	0.06	0.02	0.05	0.02

由表 3 和图 1 可以看出，粒子群算法采用八种项集贪婪修复策略都是可行的，但不同的项集贪婪修复策略在同一类数据实例上存在显著性能差异，相同的项集贪婪修复策略在不同类型实例上性能表现也有不同。同类数据的两个实例，确定平均最优近似解前三的贪心修复策略交集为该类数据较佳的算法，其中，IDKP：PSO-GGRDKP$_1$、PSO-GGRDKP$_2$，SDKP：PSO-GGRDKP$_1$、PSO-GGRDKP$_5$，WDKP：PSO-GGRDKP$_1$、PSO-GGRDKP$_8$，UDKP：PSO-GGRDKP$_2$、PSO-GGRDKP$_6$。结合每种数据实例的平均时长 T ，进一步选择各类数据实例时间性能较优的求解算法，其中，IDKP：PSO-GGRDKP$_2$，SDKP：PSO-GGRDKP$_5$， WDKP：PSO-GGRDKP$_8$， UDKP：PSO-GGRDKP$_2$。

3.3　实验二算法数据与分析

PSO-GRDKP、PSO-NGROADKP、PSO-GGRDKP 算法实验数据见表 4 所示，其中 PSO-GGRDKP 算法根据实验一分析结果，选择四类数据实例上的性能较优的贪婪修复策略进行实验。



(a) IDKP3　(b)IDKP6　(c)SDKP3　(d)SDKP6

(e)WDKP3　(f)WDKP6　(g)UDKP3　(h)UDKP6

图 1　D {0-1}KP 实例 PSO-GGRDKP 求解结果箱线图

Fig. 1　Box plots of PSO-GGRDKP results for D{0-1}KP instances

表 4　PSO-GGRDKP、PSO-NGROADKP、PSO-GRDKP 算法实验结果

Tab. 4　Algorithm experimental results of PSO-GGRDKP, PSO-NGROADKP, PSO-GRDKP

Dataset	C	*opt*	PSO-GGRDKP				PSO-NGROADKP				PSO-GRDKP			
			best	mean	worst	T /s	best	mean	worst	T /s	best	mean	worst	T /s
IDKP1	61500	70106	70090	70057	70037	1.106	70098	70097	70090	1.430	70106	70105	70098	1.284
IDKP2	103936	118268	118232	118168	118037	4.272	118235	118235	118232	5.294	118268	118268	118268	4.862
IDKP3	214453	234804	234640	234326	232619	9.216	234785	234778	234679	11.666	234802	234795	234686	11.048
IDKP4	251980	282591	282565	282350	281660	16.444	282579	282570	282423	20.327	282571	282562	282442	19.352
IDKP5	297482	335584	335460	335077	334317	29.298	335580	335562	335383	31.347	335541	335538	335474	30.543
IDKP6	415217	452463	451933	450909	449351	47.024	452410	452359	451760	47.535	452458	452412	451920	47.303
IDKP7	434677	489149	488657	487982	486639	50.667	489044	488975	488469	65.012	489118	489095	488870	58.752
IDKP8	464860	533841	532893	531437	527685	84.127	533749	533604	533060	88.726	533833	533810	533398	75.925
IDKP9	454989	528144	527458	526463	524077	92.545	528010	527978	527678	104.692	528115	528106	527965	95.705
IDKP10	496541	581244	580528	578794	576617	111.433	581043	580968	580350	142.916	581207	581194	581020	118.073
SDKP1	60143	94459	94325	93875	93098	1.178	94431	94411	94258	1.792	94232	93962	93413	1.423
SDKP2	100200	160805	160211	159881	159009	4.652	160663	160631	160359	5.340	160671	160321	159616	5.412
SDKP3	151680	238248	236996	236248	234059	12.609	238132	238081	237774	13.362	238055	237476	236394	11.994
SDKP4	214644	340027	338126	337472	336426	18.292	339817	339562	338477	21.282	339786	339055	337662	20.905
SDKP5	300632	463033	459165	457548	453168	28.383	462667	462163	461013	32.062	461309	459479	456502	33.365
SDKP6	289412	466097	462003	460618	457818	41.004	465596	465323	464135	49.359	465194	464176	462190	47.159
SDKP7	396421	620446	616378	612841	605344	56.047	619691	618860	617739	65.980	618204	616231	612674	64.845
SDKP8	424240	670697	664952	662384	654486	73.405	669310	668958	667365	83.399	668735	667212	664795	83.565
SDKP9	465564	739121	734123	730826	724267	93.362	737018	736819	735649	105.648	736476	734747	731842	107.098
SDKP10	477837	765317	758980	753068	747259	124.712	762795	762668	761977	126.717	761900	760266	757622	132.407
WDKP1	65423	83098	83025	82932	82731	1.122	83085	83082	83051	1.336	83083	83067	82927	1.361
WDKP2	102795	138215	137843	137759	137494	4.354	138175	138163	138015	5.128	138213	138185	137928	5.356
WDKP3	196603	256616	256114	255722	255289	9.431	256470	256432	256054	11.409	256586	256369	255310	12.101
WDKP4	239770	315657	314849	314640	314193	16.692	315585	315521	315022	20.221	315620	315418	314548	21.317
WDKP5	325793	428490	427124	426391	424448	26.110	428188	428068	427534	31.547	428424	428129	426928	34.034
WDKP6	351265	466050	464687	463570	461558	37.787	465756	465690	465292	46.473	465857	465599	464261	50.446
WDKP7	411219	547683	545903	544410	542740	51.114	547239	547156	546451	63.677	547369	547091	545921	70.765
WDKP8	424552	576959	574803	572924	569842	67.464	576408	576351	575937	86.628	576677	576236	574893	92.975
WDKP9	478991	650660	647825	645650	643211	85.902	650031	649728	648581	112.974	650271	649897	648347	110.999
WDKP10	493578	678967	676082	674452	670718	109.833	678113	677977	677273	137.185	678742	678413	677273	140.024
UDKP1	56429	85740	85690	85576	84893	1.282	85686	85667	85354	1.371	85676	84357	78486	1.383
UDKP2	94596	163744	163173	162263	159583	4.829	163645	163513	162327	7.232	163744	162729	156376	5.388
UDKP3	184006	269393	267595	266626	263915	10.756	268885	268721	267718	11.982	269200	266037	252792	12.004
UDKP4	204182	347599	344978	343223	337971	18.971	346991	346867	345717	25.439	347336	344506	333362	21.206
UDKP5	279856	442644	435563	432351	426821	29.374	440623	440151	438481	34.637	440938	434662	415259	33.438
UDKP6	344578	536578	529459	525276	522693	42.012	534097	532875	531044	47.067	534091	526749	503799	48.258
UDKP7	424973	635860	631620	626296	616479	57.390	632897	631970	629597	61.907	632642	625851	600343	66.428
UDKP8	412859	650206	637976	634314	626440	76.533	646661	646379	644234	84.020	645782	639408	618604	87.228
UDKP9	437234	718532	705861	700529	694344	96.546	713920	713702	712034	104.043	712556	705909	684026	113.5
UDKP10	463681	779460	762985	754952	745142	120.374	772963	772437	770927	133.303	771741	766303	746945	130.33

从表 4 看出，三类贪婪修复优化策略的粒子群算法求解质量和时间性能各不相同。定义粒子群算法的解误差率 $ERR=1-{mean}/{opt}$，各数据实例的解误差率平均值为 ERR_AVE，该值越小，算法性能越优。进一步统计三类贪婪修复优化粒子群算法四类数据实例的平均解误差率与算法平均时长见表 5 所示。表中三类贪婪修复优化粒子群算法分别是 PSO-GGRDKP, PSO-GRDKP、PSO-NGROADKP。

由表 5 看出，在四类数据实例上，三类粒子群算法的平均解误差率呈现一致趋势：$IDKP < WDKP < SDKP < UDKP$，即对 IDKP 实例求解误差率最低，而 UDKP 实例的解误差率最高。三类算法中，PSO-NGROADKP 平均解误差率最低，仅为 0.2%，PSO-GGRDKP 的平均解误差率略高于 PSO-NGROADKP、PSO-GRDKP 分别是 0.7%、0.4%。在四类数据实例上，PSO-GGRDKP 的时间性能均显著优于 PSO-GRDKP 和 PSO-NGROADKP，PSO-GGRDKP 算法时间性能较之 PSO-GRDKP 提升 12.9%，较之 PSO-NGROADKP 算法时间性能提升 13.8%。三类粒子群算法的四类数据实例时间性能趋势表现不一致，PSO-GGRDKP、PSO-NGROADKP、PSO-GRDKP 算法时间性能表现最好的数据实例各自为：WDKP、SDKP 和 IDKP。

表 5　三类粒子群算法平均解误差率与平均时长

Tab. 5　Solution average error rate and average time of three kinds of PSO

数据集	PSO-GGRDKP		PSO-NGROADKP		PSO-GRDKP	
	ERR_AVE	T_AVE	ERR_AVE	T_AVE	ERR_AVE	T_AVE
IDKP	0.002	44.613	0.000	51.895	0.000	46.285
SDKP	0.010	45.364	0.002	50.494	0.005	50.817
WDKP	0.005	40.981	0.001	51.658	0.001	53.938
UDKP	0.017	45.807	0.005	51.100	0.015	51.916
AVE	0.009	44.191	0.002	51.287	0.005	50.739

进一步探究算法解误差率与数据类型之间的关联性，表 6 给出了四类数据实例的项价值系数与项重量系数的相关系数 ρ，项价值密度均值 μ，项价值密度标准差 σ。结合表 5 统计值，易见除 SDKP 实例外，在 IDKP、WDKP、UDKP 类型上，三种算法解误差率与相关系数大小一致，即数据实例相关系数越大，则该实例上对应算法的解误差率越小。但 SDKP 相关系数略小于 IDKP、略大于 WDKP，其解误差率却高于 WDKP。此外，SDKP 与 UDKP 两类数据实例的相关系数差异很大，但其算法解误差率较为接近。从表 6 可以看出项价值密度均值有 $IDKP_\mu < WDKP_\mu < SDKP_\mu < UDKP_\mu$，项价值密度标准差有：$IDKP_\sigma < WDKP_\sigma < SDKP_\sigma < UDKP_\sigma$，这与四类数据对应求解误差率表现趋势一致，说明 D{0-1}KP 的贪婪修复优化策略粒子群算法性能与项价值密度的均值、标准差等数据特征高度相关，这进一步解释了 SDKP 与 UDKP 性能表现较为相近的情况。

表 6　D{0-1}KP 实例统计量

Tab. 6　Statistical values of D{0-1} KP instance

Data set	ρ	μ	σ
IDKP	0.956	0.844	0.222
SDKP	0.947	1.628	2.141
WDKP	0.931	1.079	0.230
UDKP	0.408	2.261	9.205

4　结束语

本文在定义 D{0-1}KP 的项集价值密度概念基础上，构造了具有八种组合策略的不可行个体贪婪修复优化方法(GGROA)，以粒子群算法为例，进一步构造了 PSO-GGRDKP 算法以探究 GGROA 方法用于求解 D{0-1}KP 的可行性和性能。D{0-1}KP 四类数据实例的算法运行结果表明：

1)适合各类实例的 GGROA 算子分别为 IDKP：PSO-GGRDKP$_2$，SDKP：PSO-GGRDKP$_5$，WDKP：PSO-GGRDKP$_8$，UDKP：PSO- GGRDKP$_2$。

2)与 PSO-NGROADKP、PSO-GRDKP 相比，PSO-GGRDKP 平均解误差率高于两个算法分别是 0.7%、0.4%，而算法时间性能较之两个算法提升 13.8%、12.9%。

3)PSO-GGRDKP、PSO-NGROADKP、PSO-GRDKP 算法解误差率与数据实例相关系数、项价值密度均值、项价值密度标准差等数据特征高度相关。

以上结果表明，PSO-GGRDKP 算法的解误差率略高于 PSO-NGROADKP、PSO-GRDKP，但较显著改善了 D{0-1}KP 的算法时间性能。

本文以粒子群算法为例，构造 D{0-1}KP 的 PSO-GGRDKP 算法并验证其性能。受各类应用 GROA 的改进启发式算法研究结果启示，本文推断在不同启发式求解算法中使用 GGROA 的求解性能可能会有差异，限于篇幅，这将作为进一步研究工作讨论。近期，文献[20]提出 D{0-1}KP 问题粒子群算法的项集个体编码方案，接下来考虑结合文献[20]的工作改进 PSO-GGRDKP 算法。

参考文献：

[1] Guder J. Discounted knapsack problems for pairs of items [D]. Nuremberg: University of Erlangen-Nurnberg, 2005.

[2] Guldan B. Heuristic and exact algorithms for discounted knapsack problems [D]. Nuremberg: University of Erlangen-Nuremberg, 2007: 1-78.

[3] 贺毅朝，王熙照，李文斌，等. 基于遗传算法求解折扣{0-1}背包问题的研究 [J]. 计算机学报, 2016, 39 (12): 2614-2630. (He Yichao, Wang Xizhao, Li Wenbin, *et al.* Research on genetic algorithms for discounted{0-1}knapsack problem [J]. China Journal of Computer, 2016, 39 (12): 2614-2630.)

[4] He Y C, Wang X Z, He Y L, *et al.* Exact and approximate algorithms for discounted{0-1}knapsack problem [J]. Information Sciences, 2016, 369 (11): 634-647.

[5] Rong A, Figueira J R, Klamroth K. Dynamic programming based algorithms for the discounted{0-1}knapsack problem [J]. Applied Mathematics & Computation, 2012, 218 (12): 6921–6933.

[6] Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems [J]. Evolutionary Computation, 1996, 4 (1): 1-32.

[7] 杨洋，潘大志，贺毅朝. 改进修复策略遗传算法求解折扣{0-1}背包问题 [J]. 计算机工程与应用, 2018, 54 (21): 37-42. (Yang Yang, PAN Dazhi, He Yichao. Improved repair strategy genetic algorithm solve discount{0-1}knapsack problem [J]. Computer Engineering and Applications, 2018, 54 (21): 37-42.)

[8] Wilbaut C, Todosijevic R, H Anafi S, *et al.* Heuristic and exact fixation-based approaches for the discounted 0-1 knapsack problem [J]. arXiv preprint, 2021, arXiv: 2106. 03438.

[9] Zhu H, He Y C, Wang X Z, *et al.* Discrete differential evolutions for the discounted{0–1}knapsack problem [J]. International Journal of Bio-Inspired Computation, 2017, 10 (4): 219-238.

[10] 吴聪聪，贺毅朝，陈嶷瑛，等. 变异蝙蝠算法求解折扣{0-1}背包问题 [J]. 计算机应用, 2017, 37 (005): 1292-1299. (Wu Congcong, He Yichao, Chen Yiying, *et al.* Mutated bat algorithm for solving discounted{0-1}knapsack problem [J]. Journal of Computer Applications, 2017, 37 (005): 1292-1299.)

[11] 冯艳红，杨娟，贺毅朝，等. 差分进化帝王蝶优化算法求解折扣{0-1}背包问题 [J]. 电子学报, 2018, 46 (6): 1343-1350. (Feng Yanhong, Yang Juan, He Yichao, *et al.* Monarch butterfly optimization algorithm with differential evolution for the discounted{0-1 knapsack problem [J]. Acta Electronica Sinica, 2018, 46 (6): 1343-1350.)

[12] Feng Y, Wang G G. Binary moth search algorithm for discounted{0-1 knapsack problem [J]. IEEE Access, 2018, 6 (99): 10708-10719.

[13] 徐小平，徐丽，王峰，刘龙. 基于Lagrange插值的学习猴群算法求解

折扣{0-1}背包问题 [J]. 计算机应用, 2020, 40 (11): 19-24. (Xu Xiaoping, Xu Li, Wang Feng, Liu Long. Learning monkey algorithm based on Lagrange interpolation to solve discounted{0-1}knapsack problem [J]. Journal of Computer Application, 2020, 40 (11): 19-24.)
[14] He Y C, Wang X Z, Gao S G. Ring theory-based evolutionary algorithm and its application to D{0-1}KP [J]. Applied Soft Computing, 2019, 77 (4): 714-722.
[15] Wu C C, Zhao J L, Feng Y H, *et al.* Solving discounted{0-1}knapsack problems by a discrete hybrid teaching-learning-based optimization algorithm [J]. Applied Intelligence, 2020, 50 (12): 1872-1888.
[16] Kennedy J, Eberhart R C. Particle swarm optimization [C]// Proceedings of IEEE International Conference On Neural Networks. Perth, Australia: IEEE, 1995: 1942-1948.
[17] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm [C]// IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, 12-15 Oct. 1997. IEEE, 1997, 5 (1): 4104–4108.
[18] Bansal J C, Deep K. A modified binary particle swarm optimization for Knapsack problems [J]. Applied Mathematics and Computation, 2012, 218 (22): 11042-11061.
[19] 麻荣永，杨磊磊，张智超. 基于粒子迭代位移和轨迹的粒子群算法 C1、C2 参数特性分析 [J]. 数学计算：中英文版, 2013, 2 (4): 109-115. (Ma Rongyong, Yang Leilei, Zhang Zhichao. Analysis the Characteristic of C1, C2 based on the PSO of Iterative Shift and Trajectory of Particle [J]. Mathematical Computation, 2013, 2 (4): 109-115)
[20] Truong T K. Different transfer functions for binary particle swarm optimization with a new encoding scheme for discounted{0-1}knapsack problem [J]. Mathematical Problems in Engineering, 2021, 2021 (4): 1-17.